Exhibit N

To

Joint Claim Chart

Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG

4th Meeting: Klagenfurt, Austria, 22-26 July, 2002

Document **JVT-D157**
File: JVT-D157.doc
Generated: 2002-08-10

Title: Joint Final Committee Draft (JFCD) of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)

Status: Approved

Contact: Thomas Wiegand
Heinrich Hertz Institute (HHI), Einsteinufer 37, D-10587 Berlin, Germany
Tel: +49 - 30 - 31002 617, Fax: +49 - 030 - 392 72 00, wiegand@hhi.de

Purpose: Report

Title page to be provided by ITU-T | ISO/IEC

DRAFT INTERNATIONAL STANDARD
DRAFT ISO/IEC 14496-10 : 2002 (E)
DRAFT ITU-T Rec. H.264 (2002 E)
DRAFT ITU-T RECOMMENDATION

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Foreword

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis. The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics. The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1. In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

ISO (the International Organisation for Standardisation) and IEC (the International Electrotechnical Commission) form the specialised system for world-wide standardisation. National Bodies that are members of ISO and IEC participate in the development of International Standards through technical committees established by the respective organisation to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organisations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75% of the national bodies casting a vote.

This Recommendation | International Standard is being submitted for approval to the ITU-T and ISO/IEC JTC1/SC29. It was prepared jointly by ITU-T SG16 Q.6 also known as VCEG (Video Coding Experts Group) and by ISO/IEC JTC1/SC29/WG11, also known as MPEG (Moving Picture Experts Group). VCEG was formed in 1997 to maintain prior ITU-T video coding standards and develop new video coding standard(s) appropriate for a wide range of conversational and non-conversational services. MPEG was formed in 1988 to establish standards for coding of moving pictures and associated audio for various applications such as digital storage media, distribution and communication.

In this Recommendation | International Standard Annexes A through E contain normative requirements and are an integral part of this Recommendation | International Standard.

0 Introduction

0.0 Prolog

As processing power and memory costs have reduced, network support for coded video data has diversified, and advances in video coding technology have progressed, the need has arisen for an industy standard for compressed video representation with substantially increased coding efficiency and enhanced robustness to network environments. Toward these ends the ITU-T video coding experts group (VCEG) and the ISO/IEC moving picture experts group (MPEG) formed a joint video team (JVT) in 2001 for development of a new ITU-T Recommendation | International Standard.

0.1 Purpose

This Recommendation | International Standard was developed in response to the growing need for higher compression of moving pictures for various applications such as video conferencing, digital storage media, television broadcasting, internet streaming and communication. It is also designed to enable the use of the coded video representation in a flexible manner for a wide variety of network environments. The use of this Recommendation | International Standard allows motion video to be manipulated as a form of computer data and to be stored on various storage media, transmitted and received over existing and future networks and distributed on existing and future broadcasting channels.

0.2 Application

This Recommendation | International Standard is designed to cover a broad range of applications for video content including but not limited to the following:

CATV	Cable TV on optical networks, copper, etc.
DBS	Direct broadcast satellite video services
DSL	Digital subscriber line video services
DTTB	Digital terrestrial television broadcasting
ISM	Interactive storage media (optical disks, etc.)
MMM	Multimedia mailing
MSPN	Multimedia services over packet networks

RTC Real-time conversational services (videoconferencing, videophone, etc.)

RVS Remote video surveillance

SSM Serial storage media (digital VTR, etc.)

0.3 Profiles and levels

This Recommendation | International Standard is designed to be generic in the sense that it serves a wide range of applications, bit rates, resolutions, qualities and services. Applications should cover, among other things, digital storage media, television broadcasting and real-time communications. In the course of creating this Specification, various requirements from typical applications have been considered, necessary algorithmic elements have been developed, and they have been integrated into a single syntax. Hence, this Specification will facilitate video data interchange among different applications.

Considering the practicality of implementing the full syntax of this Specification, however, a limited number of subsets of the syntax are also stipulated by means of "profile" and "level". These and other related terms are formally defined in clause 4.

A "profile" is a subset of the entire bitstream syntax that is defined by this Recommendation | International Standard. Within the bounds imposed by the syntax of a given profile it is still possible to require a very large variation in the performance of encoders and decoders depending upon the values taken by parameters in the bitstream such as the specified size of the decoded pictures. It is currently neither practical nor economic to implement a decoder capable of dealing with all hypothetical uses of the syntax within a particular profile.

In order to deal with this problem, "levels" are defined within each profile. A level is a defined set of constraints imposed on parameters in the bitstream. These constraints may be simple limits on numbers. Alternatively they may take the form of constraints on arithmetic combinations of the parameters (e.g. frame width multiplied by frame height multiplied by frame rate).

Coded video content conforming to this Specification uses a common syntax. In order to achieve a subset of the complete syntax, flags and parameters are included in the bitstream that signal the presence or otherwise of syntactic elements that occur later in the bitstream. In order to specify constraints on the syntax (and hence define a profile), it is thus only necessary to constrain the values of these flags and parameters that specify the presence of later syntactic elements.

0.4 Overview of the syntax

The coded representation defined in the syntax achieves a high compression capability while preserving image quality. The algorithm is not lossless as the exact sample values are not preserved through the encoding and decoding processes. Obtaining good image quality at the bit rates of interest demands very high compression, which is not achievable with intra picture coding alone. The need for random access, however, is best satisfied with pure intra picture coding. The choice of the techniques is based on the need to balance a high image quality and compression capability with the requirement to allow random access into the coded video data stream.

A number of techniques may be used to achieve high compression. The expected encoding algorithm (not specified in this Recommendation | International Standard) first uses block-based motion compensation to reduce temporal redundancy. Motion compensation is used both for causal prediction of a current picture from one or more previous pictures, and for non-causal prediction from future pictures in decoder output order. Motion vectors may be defined for a variety of region sizes in the picture. The prediction error is then further compressed using a transform to remove spatial correlation before it is quantised, producing an irreversible process that discards less important information while forming a close approximation to the source pictures. Finally, the motion vectors are combined with the quantised transform coefficient information and encoded using either variable length codes or arithmetic coding.

0.4.1 Temporal processing

Because of the conflicting requirements of random access and highly efficient compression, three main picture types are defined. Intra coded pictures (I-pictures) are coded without reference to other pictures. They provide access points to the coded sequence where decoding can begin, but are coded with only moderate compression. Inter-coded pictures (P-pictures) are coded more efficiently using motion compensated prediction of each block of sample values from some previously decoded picture selected by the encoder. Bi-predictive pictures (B-pictures) provide the highest degree of compression but require a higher degree of memory access capability in the decoding process, as each block of sample values in a B picture may be predicted using a weighted average of two blocks of motion-compensated sample values.

The organisation of the three picture types in a sequence is flexible, and the order of the decoding process is generally not the same as the order of the source picture capture process in the encoder or the output order from the decoder for display. The choice is left to the encoder and will depend on the requirements of the application. Figure Intro-1

illustrates one limited and example of the relationship among the three different picture types. Significantly different inter-picture dependency relationships are also allowed at the discretion of the encoder within limits specified by the profile and level. The decoding order is specified such that the decoding of pictures that use inter-picture prediction follows later in decoding order than other pictures that are referenced in the decoding process.

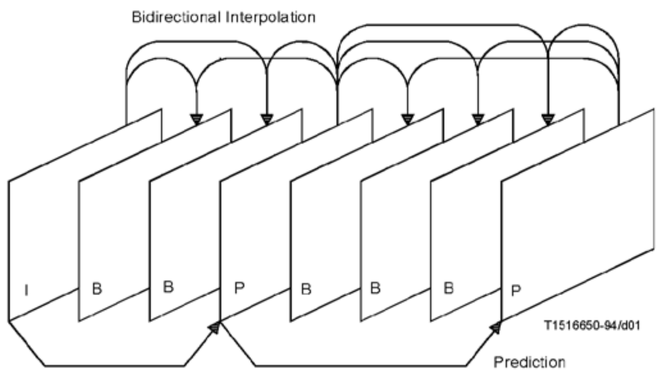


Figure Intro. 1 – Example of temporal picture structure

0.4.2 Coding interlaced video

Each frame of interlaced video consists of two fields which are separated in capture time. This Recommendation International Standard allows either the representation of complete frames or the representation of individual fields. Frame encoding or field encoding can be adaptively selected on a picture-by-picture basis and also on a more localized basis within a coded frame. Frame encoding is typically preferred when the video scene contains significant detail with limited motion. Field encoding, in which the second field can be predicted from the first, works better when there is fast movement.

0.4.3 Macroblocks and motion segmentations

As in previous video coding Recommendations and International Standard, a macroblock consisting of a 16x16 block of luma samples and a two corresponding blocks of chroma samples is used as the basic processing unit of the video decoding process.

The selection of a motion compensation unit is a result of a trade-off between the coding gain provided by using motion information and the quantity of data needed to represent it. In this Recommendation | International Standard the motion compensation process can form segmentations for motion representation as small as 4x4 in size, using motion vector accuracy of one quarter or one-eighth of a sample grid spacing displacement. The inter prediction process for motion compensated prediction of a sample block can also involve the selection of the picture to be used as the reference picture from a number of stored previously-decoded pictures.

In frame encoding, the prediction from the previous reference frame can itself be either frame-based or field-based, depending on the type of the motion vector information and other information that is encoded withing the compressed picture representation. Motion vectors are encoded differentially with respect to predicted values formed from nearby encoded motion vectors.

It is the responsibility of the encoder to calculate appropriate motion vectors or other data elements represented in the video data stream. This motion estimation process in the encoder and the selection of whether to use inter-picture prediction for the representation of each region of the video content is not specified in this Recommendation International Standard.

0.4.4 Spatial redundancy reduction

Both source pictures and prediction errors have high spatial redundancy. This Recommendation | International standard is based on the use of a block-based transform method for spatial redundancy removal. After motion compensated prediction or spatial-based prediction from previously-decoded samples within the current picture, the resulting prediction error is split into 4x4 blocks. These are converted into the transform domain where they are quantised. After quantisation many of the transform coefficients are zero or have low amplitude and can thus be represented with a small

amount of encoded data. The processes of transformation and quantization in the encoder are not specified in this Recommendation | International Standard.

1 Scope

This document specifies ITU-T Recommendation H.264 | ISO/IEC International Standard ISO/IEC 14496-10 video coding.

2 Normative references

The following Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent edition of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations.

– ITU-T Recommendation T.35 (2000), *Procedure for the allocation of ITU-T defined codes for non-standard facilities*

3 Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply.

3.1 AC coefficient: Any *transform coefficient* for which the frequency index in one or both dimensions is non-zero.

3.2 B slice: A bi-predictive *slice*; A *slice* that is coded in a manner in which a weighted average of two *inter prediction blocks* may be used for *inter prediction*.

3.3 bitstream: A sequence of bits that forms the representation of data and coded *fields* and *frames*.

3.4 block: An N-column by M-row array of samples, or NxM array of *transform coefficients*.

3.5 bottom field: One of two *fields* that comprise a *frame*. Each row of a *bottom field* is spatially located immediately below a corresponding row of a *top field*.

3.6 byte: A sequence of 8 bits, ordered from the first and most significant bit on the left to the last and least significant bit on the right.

3.7 byte aligned: A bit in a bitstream is *byte-aligned* if its position is a multiple of 8 bits from the first bit in the *bitstream*.

3.8 byte stream format: A *NAL unit stream* containing *start code prefixes* and *NAL units* as per Annex B.

3.9 category: For *slice layer* and lower layer *syntax elements*, specifies the allocation of syntax elements to data structures for *data partitioning*. It may also be used by the systems *layer* to refer to classes of syntax elements in a manner not specified in this Recommendation | International Standard.

3.10 chroma: An adjective specifying that a sample array or single sample is representing one of the two colour difference signals related to the primary colours. The symbols used for the chroma array or sample are Cr and Cb.

3.11 coded field: A coded representation of a *field*.

3.12 coded frame: A coded representation of a *frame*.

3.13 coded pictures input buffer: A first-in first-out (FIFO) buffer containing coded pictures in *decoding order* specified in the *video buffering verifier* in Annex C.

3.14 coded representation: A data element as represented in its coded form.

3.15 common intermediate format (CIF): A video frame that is 22 macroblocks wide and 18 macroblocks high.

3.16 component: An array or single sample from one of the three arrays (*luma* and two *chroma*) that make up a *field* or *frame*.

3.17 context: The numerical value of the *context variable* when decoding a *symbol*.

3.18 context modelling: The choice and specification of prior *decoded symbols* that are to be used in the *decoding* of a *symbol*.

3.19 context variable: Specified for each *symbol* by an equation containing the recently *decoded symbols* as defined by *context modelling*.

3.20 dangling field: A *field* for which there is no adjacent *field* carrying the same *frame number*.

3.21 decoding order: The order in which the *coded pictures* are to be *decoded*.

3.22 data partitioning: A method of *partitioning* selected *syntax elements* into syntactical structures based on a categorization of the *syntax elements*.

3.23 DC coefficient: The *transform coefficient* for which the frequency index is zero in both dimensions.

3.24 decoded picture: A *decoded picture* is obtained by decoding a *coded picture*. A *decoded picture* is either a *decoded frame*, or a *decoded field*. A *decoded field* is a *decoded top field* or a *decoded bottom field*.

3.25 decoded pictures buffer: A buffer specified in the *video buffering verifier* in subclause C.1. The decoded picture buffer comprises the *reference picture buffer* and the *picture reordering buffer*.

3.26 decoder: An embodiment of a *decoding process*.

3.27 decoding process: The process specified in this Recommendation | International Standard that reads a *NAL unit stream* and produces decoded *fields* or *frames*.

3.28 direct prediction: An *inter prediction* for a *block* for which no *motion vector* is decoded.

3.29 encoder: An embodiment of an *encoding process*.

3.30 emulation prevention byte: A byte having a fixed value present within a *NAL unit*. The presence of emulation prevention bytes ensures that no sequence of consecutive byte-aligned bytes in the *NAL unit* contains a *start code prefix*.

3.31 encoding process: A process, not specified in this Recommendation International Standard, that reads a sequence of *fields* and *frames* and produces a conforming *NAL unit stream* as specified in this Recommendation International Standard.

3.32 field: An assembly of alternate rows of a *frame*. A *frame* is composed of two *fields*, a *top field* and a *bottom field*.

3.33 flag: A variable which can take one of only two possible values.

3.34 frame: A *frame* contains sampled and quantized luma and chroma data of all rows of a of a video signal *frame*. A *frame* consists of two *fields*, a *top field* and a *bottom field*. For interlaced video signal, one of these *fields* is sampled temporally later than the other.

3.35 intra prediction: A *prediction* derived from the decoded samples of the same *decoded picture*.

3.36 instantaneous decoder refresh (IDR) picture: A special *I picture* that causes the *decoder* to mark all *reference pictures* in the *decoded pictures buffer* as un-used immediately before *decoding* the *IDR picture*, and to indicate that later *coded pictures* can be *decoded* without *inter prediction* from any *picture decoded* prior to the *IDR picture*.

3.37 inter coding: Coding of a *block*, *macroblock*, *slice*, or *picture* that uses information from both, within the picture and from other pictures.

3.38 inter prediction: A *prediction* derived from decoded samples of *pictures* other than the current *decoded picture*. Inter prediction is a collective term for the prediction process in P, SP, and B macroblocks.

3.39 intra coding: Coding of a *block, macroblock, slice* or *picture* that uses *intra prediction*.

3.40 I picture: An *intra picture*; A *picture* that is coded using prediction only from decoded samples within the same *picture*.

3.41 inverse transform: A part of the *decoding process* by which a *block* of *scaled transform coefficient levels* is converted into a *block* of spatial-domain samples.

3.42 layer: One of a set of syntactical structures in a non-branching hierarchical relationship. Higher layers contain lower layers. The coding layers are the picture, slice, reference picture selection, macroblock, 8x8 block and 4x4 block layers.

3.43 level: A defined set of constraints on the values which may be taken by the parameters of this Recommendation | International Standard. The same set of level definitions are used with all profiles, but individual implementations may support a different level for each supported profile. In a different context, *level* is the value of a *transform coefficient* prior to *scaling*.

3.44 **long SCP**: A *start code prefix* which is used in the construction of the *byte stream format* for *NAL unit streams*. It is mandatory at the start of a *coded picture* in *byte stream format*. Optionally it can be used instead of a *short SCP* at the start of a *coded slice* or lower *coding layers*.

3.45 **luma**: An adjective specifying that a sample array or single sample is representing the monochrome signal related to the primary colours. The symbol used for luma is Y.

3.46 **macroblock**: The 16x16 *luma* samples and the two corresponding blocks of *chroma* samples.

3.47 **macroblock address**: The *raster scan order* number of a *macroblock* starting with zero for the top left *macroblock* in a *picture*.

3.48 **macroblock allocation map**: A means of *partitioning* the *macroblocks* of a *picture* into *slice groups*. The *macroblock allocation map* an array of numbers one for each *coded macroblock* indicating the *slice group* to which the *coded macroblock* belongs.

3.49 **macroblock location**: The two dimensional coordinates of a *macroblock* in a *picture* designated by (x,y). For the top left *macroblock* of the *picture* (x,y)=(0,0). x is incremented by 1 for each *macroblock* column from left to right. y is incremented by 1 for each *macroblock* row from top to bottom.

3.50 **macroblock pair**: A pair of vertically-contiguous macroblocks in a picture that is coupled for use in macroblock-adaptive frame/field decoder processing.

3.51 **Mbit**: 1 000 000 bits.

motion compensation: Part of the *inter prediction* process for sample values, using previously decoded samples that are spatially displaced as signalled by means of *motion vectors*.

3.53 **motion vector**: A two-dimensional vector used for *motion compensation* that provides an offset from the coordinate position in the *decoded picture* to the coordinates in a *reference picture*.

3.54 **NAL unit**: A syntax structure containing an indication of the type of data to follow and bytes containing that data interspersed as necessary with *emulation prevention bytes*.

3.55 **NAL unit stream**: A sequence of *NAL units* containing the syntax structures associated with the coded video content.

3.56 **network abstraction layer (NAL)**: A definition of syntax structures and additional information including framing and timing that are supported by a system

3.57 **non-reference picture**: a *decoded picture* that is marked as not used for *inter prediction*.

3.58 **opposite parity**: The *opposite parity* of *top* is *bottom*, and vice versa.

3.59 **output order**: The order in which the *decoded pictures* are intended for output.

3.60 **output reordering delay**: A delay between decoding a *coded picture* and its output that is caused when the order of pictures specified for output is different from the order specified for decoding.

3.61 **parity**: The *parity* of a *field* can be *top* or *bottom*.

3.62 **partitioning**: The division of a set into sub-sets such that each element of the set is in exactly one of the sub-sets.

3.63 **P slice**: A *predictive slice*, A *slice* that is coded using *inter prediction* from previously-decoded *reference pictures*, using at most one *motion vector* and *reference picture index* to *predict* the sample values of each *block*.

3.64 **picture**: A collective term for a *field* or a *frame*.

3.65 **picture order count**: Picture position in output order relative to the latest IDR picture in *decoding order*.

3.66 **picture reordering**: The process of re-ordering the decoded *pictures* when the *decoding order* is different from the *output order*.

3.67 **prediction**: An embodiment of the *prediction process*.

3.68 **prediction process**: The use of a *predictor* to provide an estimate of the sample value or data element currently being decoded.

3.69 **prediction residual**: The difference between the value of a source sample or data element and its *predictor*.

3.70 **predictor**: A combination of previously decoded sample values or data elements used in the *decoding process* of subsequent sample values or data elements.

3.71 probability model: The set of probability distributions used by the arithmetic decoding process when decoding a symbol. The *context* determines which probability distribution is to be used when decoding a particular symbol at a particular point, block, macroblock, etc. in the picture. For each symbol, the number of probability distributions in the set is equal to the number of possible values for the *context variable*, i.e., the number of *contexts*.

3.72 profile: A specified subset of the syntax of this Recommendation | International Standard.

3.73 quantisation parameter: A parameter used by the *decoding process* for *scaling* of *transform coefficient levels*.

3.74 quarter common intermediate format (QCIF): A video frame that is 11 macroblocks wide and 9 macroblocks high.

3.75 random access: The ability to start the decoding of a coded NAL unit stream at a point other than the beginning of the stream and recover and exact or approximate representation of the decoded pictures represented by that NAL unit stream.

3.76 raster scan: A mapping of a rectangular two-dimensional pattern to a one-dimensional pattern such that the first entries in the one-dimensional pattern are from the first row of the two-dimensional pattern scanned from left to right, followed similarly by the second, third, etc. rows of the pattern each scanned from left to right.

3.77 reference field: A *reference field* is used for *inter prediction* when *P macroblocks*, *SP macroblocks*, and *B macroblocks* of a *coded field* or a *coded frame* are decoded.

3.78 reference frame: A *reference frame* is used for *inter prediction* when *P macroblocks*, *SP macroblocks*, and *B macroblocks* of a *coded frame* are decoded.

3.79 reference index list: A list of indices that is assigned to the *reference pictures* in the *reference picture buffer*.

3.80 reference index list 0: The list of *reference indices* for use in *reference list 0 prediction* for a *P*, *B*, or *SP slice*. All *inter prediction* used for *P* and *SP* slices is considered *reference list 0 prediction*. The *reference index list 0* is one of *two reference index lists* used for a *B slice*, with the other being the *reference index list 1*.

3.81 reference list 0 motion vector: A *motion vector* associated with a *reference index* pointing into the *reference index list 0*.

3.82 reference list 0 prediction: *Inter prediction* of the content of a *slice* using a *reference index* into the *reference index list 0*.

3.83 reference index list 1: A list of *reference indices* defined for use in *inter prediction* for a *B slice*. The *reference index list1* is one of two lists of *reference indices* used by a *B slice*, with the other being the *reference index list 0*.

3.84 reference list 1 motion vector: A *motion vector* associated with a *reference index* pointing into the *reference index list 1*.

3.85 reference list 1 prediction: *Inter prediction* of the content of a *B slice* using a *reference index* into the *reference index list 1*.

3.86 reference picture: A *picture* containing samples that are used for *inter prediction*.

3.87 reference picture buffer: A (part of the decoded pictures ?) buffer containing the *reference pictures*.

3.88 reference picture buffer management: specifies in the coded data, how the *decoding process* applies to the *decoded pictures buffer* and in particular to the *reference picture buffer*.

3.89 reserved: The term "reserved", when used in the clauses defining some values of a particular syntax element means that these values may be used in extensions of this Recommendation | International Standard by ITU-T | ISO/IEC, and that these values shall not be used unless so specified.

3.90 residual: The decoded difference between a *prediction* of a sample or data element and its decoded value.

3.91 run: A number of consecutive data elements represented in the decoding process. In one context, the number of zero-valued *transform coefficients* preceding a non-zero *transform coefficient*, in the *block scan* order. In another context, the number of *skipped macroblocks*.

3.92 sample aspect ratio (SAR): Specifies the distance between *luma* samples. It is defined as the vertical displacement of the rows of *luma* samples in a *frame* divided by the horizontal displacement of the *luma* samples. Thus its units are (metres per row) ÷ (metres per sample).

3.93 scaling: The process of *scaling* the *transform coefficient levels* resulting in *transform coefficients*.

3.94 short SCP: A *start code prefix* which is used in the construction of a *byte stream format* of coded data. It can be used instead of a *long SCP* at the start of a *coded slice* or lower *coding layers*.

3.95 **skipped macroblock**: A *macroblock* for which no data is coded other than an indication that the *macroblock* is to be decoded as "skipped". This indication may be common to several *macroblocks*.

3.96 **slice**: An integer number of *macroblocks* ordered contiguously in *raster scan order* within a particular *slice group*. Although a slice contains macroblocks that are contiguous in raster scan order within a slice group, these macroblocks are not necessarily contiguous within the picture. The addresses of the *macroblocks* are derived from the address of the first macroblock and the *slice group* parameters.

3.97 **slice group**: A sub-set of the *macroblocks* of a *picture*. The division of the *picture* into slice groups is a *partitioning* of the *picture*. The partition is specified by the *slice group* parameters.

3.98 **slice header**: A part of a *coded slice* containing the *coded representation* of data elements pertaining to the *slice data* that follow the *slice header*.

3.99 **SI picture**: A *switching I picture*; A *picture* that is coded using prediction only from decoded samples within the same *picture*, encoded such that it can be reconstructed identically to another *SP slice* or *SI slice*, as specified in clause 11.

3.100 **SI slice**: A *switching I slice*; A *slice* that is coded using prediction only from decoded samples within the same *slice*, encoded such that it can be reconstructed identically to another *SP slice* or *SI slice*, as specified in subclause 11.

3.101 **source (input)**: Term used to describe the video material or some of its attributes before encoding.

3.102 **SP slice**: A *switching P slice*; A *slice* that is coded using *inter prediction* from previously-decoded *reference pictures*, using at most one *motion vector* and *reference picture index* to *predict* the sample values of each *block*, encoded such that it can be reconstructed identically to another *SP slice* or *SI slice*, as specified in subclause 11.

3.103 **start code prefix (SCP)**: One of a set of unique codes embedded in the *byte-stream format* that are used for identifying the beginning of a coding layer. Emulation of *start code prefixes* is prohibited within NAL units.

3.104 **string of data bits (SODB):** An ordered sequence of some finite number of bits, in which the left-most bit is considered to be the first and most significant bit (MSB) and the right-most bit is considered to be the last and least significant bit (LSB).

3.105 **symbol**: A *syntax element*, or part thereof, to be decoded.

3.106 **top field**: One of two *fields* that comprise a *frame*. Each row of a *top field* is spatially located immediately above the corresponding row of the *bottom field*.

3.107 **transform coefficient**: A scalar considered to be in a frequency domain that is associated with a particular two-dimensional frequency index in the *inverse transform* of the decoding process.

3.108 **variable length coding (VLC)**: A reversible procedure for entropy coding that assigns shorter code-words to frequent *symbols* and longer code-words to less frequent *symbols*.

3.109 **video buffering verifier (VBV)**: A hypothetical *decoder* that is connected to the *output* of the *encoder*. Its purpose is to provide a constraint on the variability of the NAL unit stream that an encoder or editing process may produce.

3.110 **XYZ profile decoder**: A decoder able to decode coded data conforming to the specifications of the XYZ profile (with XYZ being any of the defined Profile names).

3.111 **zig-zag scan**: A specific sequential ordering of *transform coefficients* from (approximately) the lowest spatial frequency to the highest.

4 Abbreviations

4.1 **ABT**: Adaptive Block size Transform

4.2 **CABAC**: Context-based Adaptive Binary Arithmetic Coding

4.3 **CAVLC**: Context-based Adaptive Variable Length Coding

4.4 **CIF**: Common Intermediate Format

4.5 **DPA**: Data Partition type A

4.6 **DPB**: Data Partition type B

4.7 **DPC**: Data Partition type C

4.8 **FCC**: Federal Communications Commission

4.9 **FIFO**: First-In, First-Out

4.10 **IDR**: Instantaneous Decoder Refresh

4.11 **LPS**: Least Probable Symbol

Formatted: Font: Not Bold

4.12 **LSB**: Least Significant Bit

4.13 **MB**: Macroblock

4.14 **MPS**: Most Probable Symbol

Formatted: Font: Not Bold

4.15 **MSB**: Most Significant Bit

4.16 **NAL**: Network Abstraction Layer

4.17 **QCIF**: Quarter Common Intermediate Format

4.18 **RBSP**: Raw Byte Sequence Payload

4.19 **SAR**: Sample Aspect Ratio

4.20 **SCP**: Start Code Prefix

4.21 **SEI**: Supplemental Enhancement Information

4.22 **SMPTE:** Society of Motion Picture and Television Engineers

4.23 **SODB**: String Of Data Bits

4.24 **VCL**: Video Coding Layer

4.25 **VBV**: Video Buffering Verifier

4.26 **VLC**: Variable Length Coding

5 Conventions

The mathematical operators used to describe this Specification are similar to those used in the C programming language. However, integer divisions with truncation and rounding are specifically defined. Numbering and counting loops generally begin from zero.

5.1 Arithmetic operators

The following mathematical and logical operators are defined as follows

+	Addition
–	Subtraction (as a binary operator) or negation (as a unary operator)
++	Increment, i.e. *x*++ is equivalent to x = x + 1
––	Decrement, i.e. *x*–– is equivalent to x = x – 1
×;* }	Multiplication
^	Power
/	Integer division with truncation of the result toward zero. For example, 7/4 and –7/–4 are truncated to 1 and –7/4 and 7/–4 are truncated to –1.
DIV	Integer division with truncation of the result toward minus infinity. For example 3 DIV 2 is rounded to 1, and –3 DIV 2 is rounded to –2.
÷	Used to denote division in mathematical equations where no truncation or rounding is intended.
$\sum_{i=a}^{b} f(i)$	The summation of the $f(i)$ with i taking all integer values from a up to and including b.
a % b	Modulus operator. Remainder of a divided by b, defined only for a and b both positive integers

5.2 Logical operators

a && b	Boolean logical "and" of a and b
a || b	Boolean logical "or" of a and b

!	Logical NOT

5.3 Relational operators

>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
==	Equal to
!=	Not equal to

5.4 Bit-wise operators

&	AND
|	OR
$a >> b$	Arithmetic right shift of a two's complement integer representation of a by b binary digits. This function is defined only for positive values of b. Bits shifted into the MSBs as a result of the right shift shall have a value equal to the MSB of a prior to the shift operation.
$a << b$	Arithmetic left shift of a two's complement integer representation of a by b binary digits. This function is defined only for positive values of b.

5.5 Assignment

=	Assignment operator

5.6 Functions

$$\text{Sign}(\,x\,) = \begin{cases} 1 & ; \quad x \geq 0 \\ -1 & ; \quad x < 0 \end{cases} \tag{5-1}$$

$$\text{Abs}(\,x\,) = \begin{cases} x & ; \quad x \geq 0 \\ -x & ; \quad x < 0 \end{cases} \tag{5-2}$$

$$\text{Clip3}(\,a, b, c\,) = \begin{cases} a & ; \quad c < a \\ b & ; \quad c > b \\ c & ; \quad \text{otherwise} \end{cases} \tag{5-3}$$

$$\text{Clip1}(\,x\,) = \text{Clip3}(\,0, 255, x\,) \tag{5-4}$$

Ceil(x) rounds x up to the nearest integer. Defined only for non-negative values of x. (5-5)

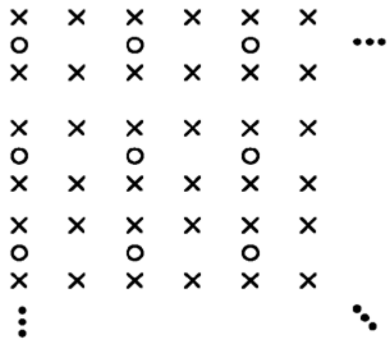
Log2(x) returns the base-2 logarithm of x. (5-6)

6 Source coder

6.1 Picture formats

The image width and height of the decoded luma memory arrays are multiples of 16 samples. Decoder output picture sizes that are not a multiple of 16 in width or height can be specified using a cropping rectangle. This Recommendation International Standard represents colour sequences using 4:2:0 chroma sampling.

The nominal vertical and horizontal locations of luma and chroma samples in frames are shown in Figure 6-1. Alternative chroma sample locations may be indicated in video usability information syntax (see Annex E).



Guide:
× = Location of luminance sample
O = Location of chrominance sample

Figure 6-1 – Nominal vertical and horizontal locations of 4:2:0 luma and chroma samples in a frame

This Recommendation | International Standard describes decoding of video that contains either progressive-scan or interlaced-scan frames, which may be mixed together in the same sequence.

A decoded frame of video contains two fields, the top field and the bottom field, which are interleaved. The first (i.e., top), third, fifth, etc. rows of a decoded frame are the top field rows. The second, fourth, sixth, etc. rows of a decoded frame are the bottom field rows. A top field picture consists of only the top field rows of a decoded frame. A bottom field picture consists of only the bottom field rows of a decoded frame.

The two decoded fields of an interlaced frame are separated in time. They may be decoded separately as two fields or together as a frame.

NOTE - A progressive frame should always be coded as a single frame picture. However, a progressive frame is still considered to consist of two fields (at the same instant in time).

The nominal vertical and horizontal locations of luma and chroma samples in interlaced frames are shown in Figure 6-2. The vertical sampling locations of the chroma samples in a top field of an interlaced frame are specified as shifted up by 1/4 luma sample height relative to the field-sampling grid in order for these samples to align vertically to the usual location relative to the full-frame sampling grid. The vertical sampling locations of the chroma samples in a bottom field of an interlaced frame are specified as shifted down by 1/4 luma sample height relative to the field-sampling grid in order for these samples to align vertically to the usual location relative to the full-frame sampling grid. The horizontal sampling locations of the chroma samples are specified as unaffected by the application of interlaced field coding.

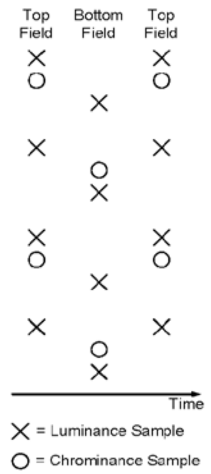


Figure 6-2 – Nominal vertical and temporal sampling locations of samples in 4:2:0 interlaced frames

6.2 Spatial subdivision of a picture into macroblocks

Pictures are divided into macroblocks. For instance, a QCIF picture is divided into 99 macroblocks as indicated in Figure 6-3.

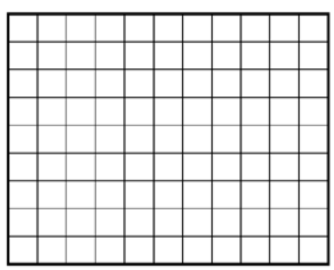


Figure 6-3 – A picture with 11 by 9 macroblocks (QCIF picture)

6.3 Calculation of the macroblock address

When mb_adaptive_frame_field_flag in the picture parameter set is 0, the macroblock address calculation is recursively specified as follows:

1. A coded slice contains in its slice header the macroblock address of the first macroblock in the coded slice. The macroblock allocation map conveys the slice group identifier of the first macroblock in a slice.

2. Let g be the slice group identifier of the most recently decoded macroblock of a given coded slice. The next macroblock address is found by searching the macroblock allocation map in scan order for the next macroblock that has the same slice group identifier g.

When mb_adaptive_frame_field_flag in the picture parameter set is 1, the macroblock pair address calculation is recursively specified as follows (see Figure 6-4):

1. A coded slice contains in its slice header the macroblock pair address of the first macroblock pair in the coded slice. The slice group index of the first macroblock pair in a coded slice may be found by referencing the macroblock allocation map.

2. Let g be the slice group identifier of the most recently decoded macroblock pair of a given coded slice. The next macroblock pair address is found by searching the macroblock allocation map in scan order for the next macroblock pair that has the same slice group identifier g.

NOTE - This note describes one of many possible implementations of the macroblock address calculation (the macroblock pair address could be calculated with a similar algorithm).
Assume the availability of a one-dimensional array m with as many entries as there are macroblocks or macroblock pairs in the picture, and that is initialized with a one-dimensional representation of the macroblock allocation map of the picture parameter set (in one of the several explicit or implicit forms defined there). Let n be the macroblock address of the last decoded macroblock of a given slice. The next macroblock address is calculated by the following three steps:

1. Identify the slice group of the macroblock n by using n as an index into m.
2. Search in m in ascending order, starting with n, for the next entry that has the same slice group identifier.
3. This is the macroblock address for the next macroblock or macroblock pair of the coded slice.

NOTE - a coded slice may consist of one slice NAL unit or, when data partitioning is used, of three NAL units DPA, DPB, and DPC.

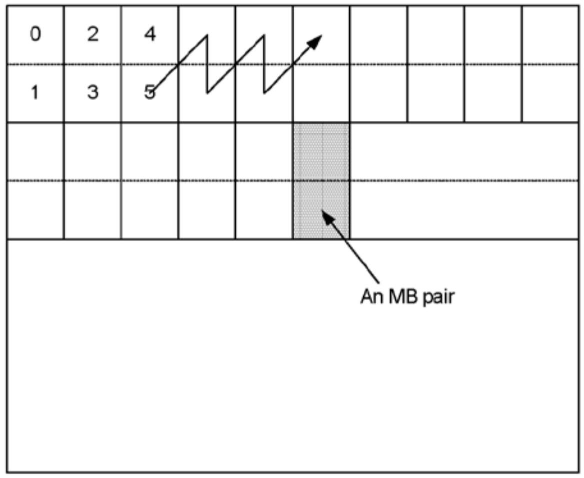


Figure 6-4 – Partitioning of the decoded frame into macroblock pairs. An MB pair can be coded as two frame MBs, or one top-field MB and one bottom-field MB. The numbers indicate the scanning order of coded MBs.

6.4 Assignment of symbols within a macroblock

Figures 6-5 indicates how a macroblock or sub macroblock is partitioned with each luma block and associated chroma blocks being motion-compensated using a separate motion vector and (for luma blocks larger or equal to 8x8 samples and associated chroma blocks) using a separate reference picture index. If the ABT feature is used, the transform for residual coding is adapted to the partitioning pattern as well.

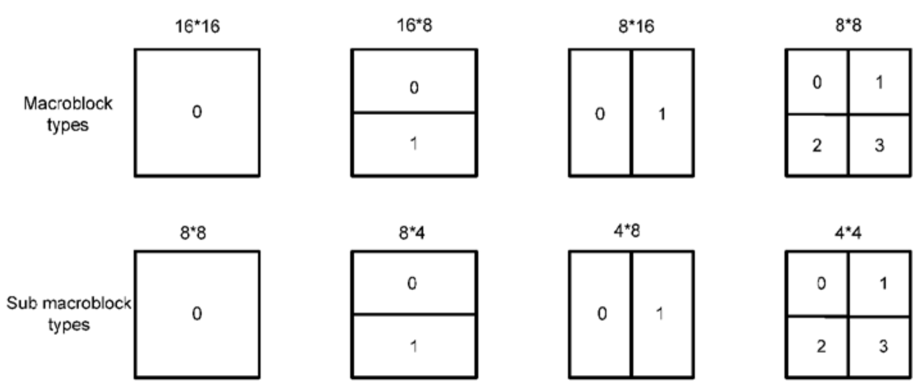


Figure 6-5 – Numbering of the vectors for the different blocks in raster scan order depending on the inter mode. For each block the horizontal component comes first followed by the vertical component.

Figure 6-6 shows the order of the assignments of syntax elements resulting from coding a macroblock to sub-blocks of the macroblock if the ABT feature is not used. The assignment order if the ABT feature is used is specified in Figure 12-1.

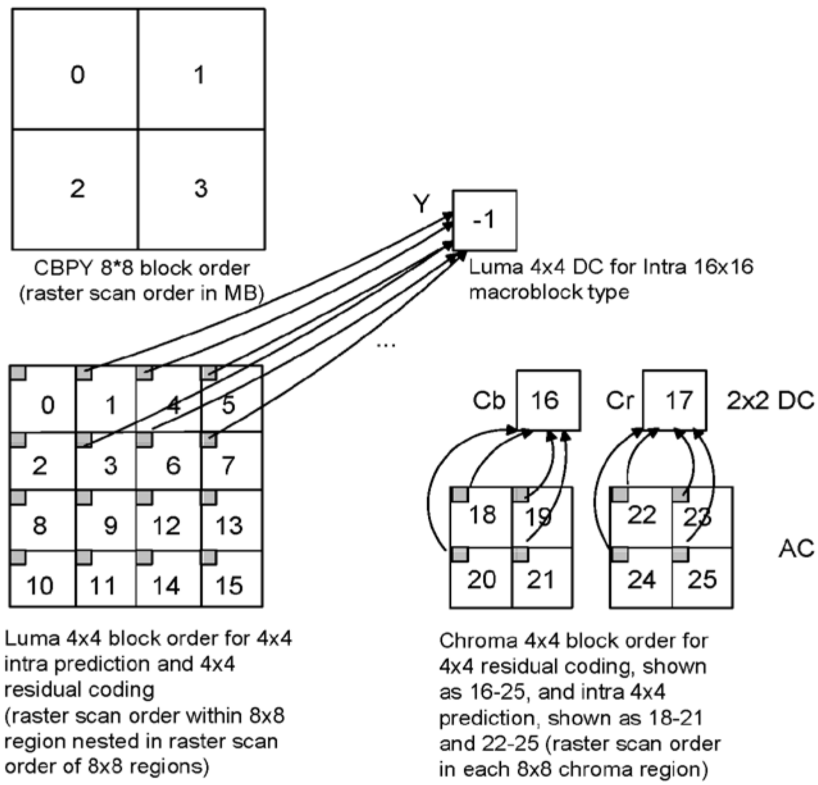


Figure 6-6 – Ordering of blocks for coded_block_patternY, 4x4 intra prediction, and 4x4 residual coding

7 Syntax and semantics

7.1 Method of describing the syntax in tabular form

The syntax is described in a manner that closely follows the C-language syntactic constructs. Syntax elements in the bitstream are represented in **bold** type. Each syntax element is described by its name, its syntax category and descriptor for its method of coded representation. A decoder behaves according to the value of the syntax element and on the values of previously decoded syntax elements.

The syntax tables describe a superset of the syntax of all correct and error-free input bitstreams. Additional constraints on the syntax form may be specified in other clauses. An actual decoder must implement correct means for identifying entry points into the bitstream for proper decoding and to identify and handle errors in the bitstream. The methods for identifying and handling errors and other such situations are not described here.

Following C-language conventions, a value of '0' represents a FALSE condition in a test statement. The value TRUE is represented by '1', but any other value different than zero is also understood as TRUE.

The following table lists examples of pseudo code used to describe the syntax. When syntax_element appears, it indicates that a data element is read (extracted) from the bitstream and the bitstream pointer advances to the bit following the last bit of the data element extracted.

	Category	Descriptor
/* A statement can be a syntax element with an associated syntax category and descriptor or can be an expression used to specify conditions for the existence, type, and quantity of syntax elements, as in the following two examples */		
syntax_element	3	e(v)
conditioning statement		
/* A group of statements enclosed in curly brackets is a compound statement and is treated functionally as a single statement. */		
{		
statement		
statement		
…		
}		
/* A "while" structure indicates a test of whether a condition is true, and if true, indicates evaluation of a statement (or compound statement) repeatedly until the condition is no longer true */		
while(condition)		
statement		
/* A "do … while" structure indicates evaluation of a statement once, followed by a test of whether a condition is true, and if true, indicates repeated evaluation of the statement until the condition is no longer true */		
do		
statement		
while(condition)		
/* An "if … else" structure indicates a test of whether a condition is true, and if the condition is true, indicates evaluation of a primary statement, otherwise indicates evaluation of an alternative statement. The "else" part of the structure and the associated alternative statement is omitted if no alternative statement evaluation is needed */		
if(condition)		
primary statement		
else		
alternative statement		
/* A "for" structure indicates evaluation of an initial statement, followed by a test of a condition, and if the condition is true, indicates repeated evaluation of a primary statement followed by a subsequent statement until the condition is no longer true. */		
for(initial statement; condition; subsequent statement)		
primary statement		

7.2 Definitions of functions and descriptors

The functions presented here are used in the syntactical description. These functions assume the existence of a bitstream pointer with an indication of the position of the next bit to be read by the decoder from the bitstream.

byte_aligned( )

- Returns TRUE if the current position in the bitstream is on a byte boundary, i.e., the next bit in the bitstream is the first bit in a byte. Otherwise it returns FALSE

first_non_skip_mb_in_pair( )

- Returns TRUE if the current macroblock is the first macroblock in a macroblock pair or if the previous macroblock in the macroblock pair was skipped. Used only in macroblock-adaptive frame/field coding.

Formatted: Bullets and Numbering

next_bits(*n*)

- Provides the next bits in the bitstream for comparison purposes, without advancing the bitstream pointer. Provides a look at the next *n* bits in the bitstream with *n* being its argument. If used within an RBSP syntax structure or in a structure within an RBSP syntax structure, returns a non-matching value if fewer than *n* bits remain within the RBSP prior to the rbsp_trailing_bits( ). If used within the byte stream format syntax specified in Annex B, returns a non-matching value if fewer than *n* bits remain within the byte stream.

more_rbsp_data( )

- Returns TRUE if there is more data in an RBSP before rbsp_trailing_bits( ). Otherwise it returns FALSE. The method for enabling determination of whether there is more data in the slices is specified by the system (or in Annex B for systems that use the byte stream format).

~~num~~total_coeff( )

- Returns the number of coefficients ~~(minus trailing ones)~~ from coeff_token. See subclause 7.3.5.3.1.

Formatted: Default Paragraph Font

trailing_ones( )

- Returns the trailing ones from coeff_token. See subclause 7.3.5.3.1.

Formatted: Default Paragraph Font

slice_type( )

- Returns the coding type of slice.

The following descriptors are used to describe the type of each syntax element.

- **b(8):** byte having any value (8 bits).
- **ue(v)**: unsigned integer Exp-Golomb-coded syntax element with the left bit first.
- **se(v)**: signed integer Exp-Golomb-coded syntax element with the left bit first.
- **me(v)**: mapped Exp-Golomb-coded syntax element with the left bit first.
- **ce(v)**: context-adaptive variable-length entropy-coded syntax element with the left bit first.
- **f(n)**: fixed-value bit string using *n* bits written (from left to right) with the left bit first.
- **i(n)**: signed integer using *n* bits for a two's complement representation with most significant bit written first. If *n* is "v", the number of bits varies in a manner dependent on the value of other decoded data.
- **u(n)**: unsigned integer using *n* bits with most significant bit written first. If *n* is "v", the number of bits varies in a manner dependent on the value of other decoded data.
- **xe(v)**: extended Exp-Golomb-coded syntax element with left bit first. If indicating a selection from a list having only two alternatives, shall be interpreted as u(1). If indicating a selection from a list having more than two alternatives, shall be interpreted as ue(v).
- **ac(v)**: context-adaptive arithmetic entropy-coded syntax element. Some syntax elements are coded using CABAC when entropy_coding_mode = = 1. This is indicated by specifiying an alternative descriptor separated by a bar.

7.3 Syntax in tabular form

7.3.1 NAL unit syntax

nal_unit(NumBytesInNALunit) {	Category	Descriptor
forbidden_bit		u(1)
nal_storage_idc		u(2)
nal_unit_type		u(5)
NumBytesInRBSP = 0		
for(i = 0; i < NumBytesInNALunit-1; i++) {		
if(next_bits(16) = = 0x0003) {		
rbsp[NumBytesInRBSP++ **]**		b(8)
i++		
emulation_prevention_byte /* = = 0x03 */		f(8)
} else		
rbsp[NumBytesInRBSP++ **]**		b(8)
}		
}		

7.3.2 Raw byte sequence payloads and RBSP trailing bits syntax

7.3.2.1 Sequence parameter set RBSP syntax

seq_parameter_set_rbsp( ) {	Category	Descriptor
profile_idc	0	ue(v)
level_idc	0	ue(v)
seq_parameter_set_id	0	ue(v)
log2_max_frame_num_minus4	0	ue(v)
pic_order_cnt_type	0	ue(v)
if(pic_order_cnt_type = = 0)		
log2_max_pic_order_cnt_minus4	0	ue(v)
else if(pic_order_cnt_type = = 1) {		
offset_for_non_stored_pic	0	se(v)
num_stored_frames_in_pic_order_cnt_cycle	0	ue(v)
for(i = 0; i < num_stored_frames_in_pic_order_cnt_cycle; i++)		
offset_for_stored_frame	0	se(v)
}		
num_of_ref_frames	0	ue(v)
required_frame_num_update_behaviour	0	u(1)
pic~~frame~~_width_in_mbs_minus1	0	ue(v)
pic~~frame~~_height_in_mbs_minus1	0	ue(v)
send_filter_parameters_flag	0	u(1)
constrained_intra_pred_flag	0	u(1)
mb_frame_field_adaptive_flag	0	u(1)
vui_seq_parameters_flag	0	u(1)
if(vui_seq_parameters_flag)		
vui_seq_parameters( )	0	
rbsp_trailing_bits( )	0	
}		

7.3.2.2 Picture parameter set RBSP syntax

pic_parameter_set_rbsp( ) {	Category	Descriptor
pic_parameter_set_id	1	ue(v)
seq_parameter_set_id	1	ue(v)
entropy_coding_mode	1	ue(v)
motion_resolution	1	ue(v)
adaptive_block_size_transform_flag	1	u(1)
num_slice_groups_minus1	1	ue(v)
if(num_slice_groups_minus1 > 0) {		
mb_allocation_map_type	1	ue(v)
if(mb_allocation_map_type == 0)		
for(i = 0; i <= num_slice_groups_minus1; i++)		
run_length	1	ue(v)
else if(mb_allocation_map_type == 2)		
for(i = 0; i < num_mbs_in_pic; i++)		
slice_group_id	1	u(v)
else if(mb_allocation_map_type == 3)		
for(i = 0; i < num_slice_groups_minus1; i++) {		
top_left_mb	1	u(v)
bottom_right_mb	1	u(v)
}		
else if(mb_allocation_map_type == 4 || mb_allocation_map_type == 5 || mb_allocation_map_type == 6) {		
slice_group_change_direction	1	u(1)
slice_group_change_rate_minus1	1	ue(v)
}		
}		
num_ref_idx_l0_active_minus1	1	ue(v)
num_ref_idx_l1_active_minus1	1	ue(v)
weighted_pred_flag	1	u(1)
weighted_bipred_explicit_flag	1	u(1)
weighted_bipred_implicit_flag	1	u(1)
slice_qp_minus26 /* relative to 26 */	1	se(v)
slice_qp_s_minus26 /* relative to 26 */	1	se(v)
redundant_slice_flag	1	u(1)
vui_pic_parameters_flag	1	u(1)
if(vui_pic_parameters_flag) {		
vui_pic_parameters( )	1	
}		
rbsp_trailing_bits( )	1	
}		

7.3.2.3 Supplemental enhancement information RBSP syntax

sei_rbsp( ) {	Category	Descriptor
do		
sei_message( )	7	
while(more_rbsp_data( ))		
rbsp_trailing_bits( )	7	
}		

7.3.2.3.1 Supplemental enhancement information message syntax

sei_message( ) {	Category	Descriptor
PayloadType = 0		
while(next_bits(8) = = 0xFF) {		
byte_ff /* equal to 0xFF */	7	u(8)
PayloadType += 255		
}		
last_payload_type_byte	7	u(8)
PayloadType += last_payload_type_byte		
PayloadSize = 0		
while(next_bits(8) = = 0xFF) {		
byte_ff	7	u(8)
PayloadSize += 255		
}		
last_payload_size_byte	7	u(8)
PayloadSize += last_payload_size_byte		
sei_payload(PayloadType, PayloadSize)	7	
}		

7.3.2.4 Picture delimiter RBSP syntax

pic_delimiter_rbsp( ) {	Category	Descriptor
three_reserved_bits	8	u(3)
pic_type	8	u(3)
non_stored_pic_flag	8	u(1)
rbsp_trailing_bits( )	8	
}		

7.3.2.5 Filler data RBSP syntax

filler_data_rbsp(NumBytesInRBSP) {	Category	Descriptor
while(next_bits(8) = = 0xFF)		
byte_ff	9	f(8)
rbsp_trailing_bits( )	9	
}		

7.3.2.6 Slice layer RBSP syntax

slice_layer_no_partitioning_rbsp( ) {	**Category**	**Descriptor**
slice_header( )	4	
slice_data( ) /* all categories of slice_data( ) syntax */	4 | 5 | 6	
rbsp_slice_trailing_bits( )	4	
}		

7.3.2.7 Data partition RBSP syntax

7.3.2.7.1 Data partition A RBSP syntax

dpa_layer_rbsp( ) {	**Category**	**Descriptor**
slice_header( )	4	
slice_id	4	ue(v)
slice_data( ) /* only the category 4 parts of slice_data( ) syntax */	4	
rbsp_slice_trailing_bits( )	4	
}		

7.3.2.7.2 Data partition B RBSP syntax

dpb_layer_rbsp( ) {	**Category**	**Descriptor**
slice_id	5	ue(v)
slice_data( ) /* only the category 5 parts of slice_data( ) syntax */	5	
rbsp_slice_trailing_bits( )	5	
}		

7.3.2.7.3 Data partition C RBSP syntax

dpc_layer_rbsp( ) {	**Category**	**Descriptor**
slice_id	6	ue(v)
slice_data( ) /* only the category 6 parts of slice_data( ) syntax */	6	
rbsp_slice_trailing_bits( )	6	
}		

7.3.2.8 RBSP trailing bits syntax

rsbp_trailing_bits( ) {	**Category**	**Descriptor**
rbsp_stop_bit /* equal to 1 */	All	f(1)
while(!byte_aligned( ))		
rbsp_alignment_bit /* equal to 0 */	All	f(1)
}		